Software Security Requirements Management as an Emerging Cloud Computing Service

Muthu Ramachandran

School of Computing, Creative Technologies and Engineering
Leeds Beckett University
Leeds LS6 3QS UK
Email: M.Ramachandran@leedsbeckett.ac.uk

Abstract
Emerging cloud applications are growing rapidly and the need for identifying and managing service requirements is also highly important and critical at present. Software Engineering and Information Systems has established techniques, methods and technology over two decades to help achieve cloud service requirements, design, development, and testing. However, due to the lack of understanding of software security vulnerabilities that should have been identified and managed during the requirements engineering phase, we have not been so successful in applying software engineering, information management, and requirements management principles that have been established for the past at least 25 years, when developing secure software systems. Therefore, software security cannot just be added after a system has been built and delivered to customers as seen in today's software applications. This paper provides concise methods, techniques, and best practice requirements management guidelines for software security and also discusses an Integrated-Secure SDLC model (IS-SDLC), which will benefit practitioners, researchers, learners, and educators. This paper illustrates our approach for a large cloud system Amazon EC2 service.

Keywords: Emerging cloud services**, S**oftware Security Engineering, Software security requirements engineering, Secured Software Development, SQUARE method, BSI, Touchpoint, SDL, Requirements Management

1 Introduction

There is no doubt that the cloud computing has revolutionised human lives, communications, digital economy, socialisation, and entertainment. At the same time demands for internet enabled applications grows rapidly. Almost all businesses, applications, entertainment devices, mobile devices, robots, large scale systems (aircrafts, mission control systems), safety-critical systems, medical systems, internet of things devices are internet enabled for various reasons such as online upgrade, distributed applications, team projects, and server connectivity. Therefore, there is ever growing demand for secured applications and trust. Cyber attacks are increasing continuously From spam, phishing, identify theft, and others in much larger scale attacks such as money laundering and cyber terrorism. There is a real possibility that a cyber attack could disable command systems, bring down power grids, open dam floodgates, paralyses communications and transport systems, creating mass hysteria: Any or all of which could be the precursor to terrorist or military attack. These are some of the threats since we (personal, govt. organisations, companies, and business) mostly depend on computers and mobiles for communications and management.

Emerging cloud services are on the increase including eHealth Cloud, E-Learning, E-Manufacturing, etc. Kostoska, M, Gusev, M, and Ristov, S (2014) describe a new cloud protability platform (PaaS) as a service which can accept and exchange from one cloud platform to another platform and installed completely automatically. Han, G et al (2016) have proposed an energy-aware VM consolidation based on remaining utilisation-aware algorithm (RUA). Xu (2013) has proposed an interoperable cloud manufacturing system (ICMS) which shares encapsulated resources into a cloud service for manufacturing where manufacturing capabilities and business opportunities are integrated and broadcasted in a larger resource pool. Shrivastava et al. (2015) describes how eHealth initiatives and the technology helps to achieve health services at a very low cost by connecting and consulting expertise worldwide

for very complicated surgeries, use cloud services, IoT services for collecting and monitoring data, and they also recommend to integrate various technologies.

The cloud vendors include well known businesses such as Amazon EC2, IBM, HP. Google, Microsoft Azure. In addition, there are new cloud businesses on the market such as 2nd Watch which has partnered with AWS (Amazon EC2 Web Service) offers cloud migration, workload management, and has has 75,000 instances from hundreds of customers under its management. Similarly, BetterCloud automated management and data security for cloud office platforms, including Google Apps and Microsoft Office 365 (Whiting 2015) Hsu and Cheng (2015) describe a Semantic Agent as a Service (SAaaS) which collects and discover knowledge from semantic information and works with core cloud services: SaaS, PaaS, and IaaS. All of the cloud vendors agree the strong need for secure cloud services and business migration.

This paper aims to outline the importance of developing secure cloud services using a disciplined approach known as software security engineering and it is also known as secure software development. In particular, this paper identifies key methods and techniques on software security requirements engineering as it is the heart of developing secure cloud services. This paper discusses clear best practice guidelines on software security and discusses our Integrated-Secure SDLC (IS-SDLC) model which overcomes current difficulties in identifying and visually representing security process which have been elaborated From security requirements. In addition, we all enjoy the new technologies based on service computing, such as mobile apps, cloud storages, social media networks, and cloud services such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). The need for high performance cloud computing and accuracy and precision of big data enforce the need for complete identification of non-functions cloud service requirements and its specification. In addition, security, privacy & trust are the key to the success of cloud computing, which needs to be identified and specified as part of the requirements management process. This paper divided into: section 1 discusses on introduction to our work, section 2 provides the rationale for the subject area, section 3 introduces software security requirements engineering and management, and section 4 introduces an integrated approach to SDLC, and the final section provides a large scale case study using Microsoft Security Development Lifecycle on threat modelling techniques for Amazon EC2 cloud services.

2 Why Software Security Engineering?

Software Engineering has established techniques, methods, and technology over two decades. However, security issues are direct attributes of various software such as applications, user interface, networking, distribution, data-intensive transactions, and communication tools, etc. Current applications are being developed and delivered where security has been patched as aftermath. Early commercial developers have tackled security problems using firewalls (at the application level), penetration testing, and patch management.

We are also faced with tackling fast growing information warfare, cybercrime, cyber-terrorism, identify theft, spam, and other various threats. Therefore, it is important to understand the security concerns starting From requirements, design, and testing to help us Build-In Security (BSI) instead of batching security afterwards. McGraw (2006) says *a central and critical aspect of the computer security problem is a software problem*. This paper defines *software security engineering as a discipline which considers capturing and modelling for security, design for security, adopting best practices, testing for security, managing, and educating software security to all stakeholders*.

Software engineering has well established framework of methods, techniques, rich processes that can address small to very large scale products and organisations (CMM, CMMi, SPICE, etc.), and the associated technology such as modelling (UML), CASE tools, and CAST tools, and others. Software Engineering has also been well established quality models and methods, reuse models and methods, reliability models and methods, and numerous lists of other techniques. The so called -lities of software engineering long has been contributed as part of quality attributes (Quality, Testability, Maintainability, Security, Reliability, Reusability). These attributes can't be just added on to the system as they have to be built in From early part of the life cycle stages (a typical software development lifecycle include starting From requirements engineering (RE), software specification, software & architectural design, software development (coding), software testing, and maintenance. Security has become highly important attribute since the development of online based applications. Software project management has well established techniques and breadth of knowledge including global development (due to emergence of internet

revolution and people skills across the globe), cost reduction techniques, risk management techniques, and others. Nowadays, most of the current systems and devices are web enabled and hence security needs to be achieved right From beginning: need to be identified, captured, designed, developed and tested. Ashford (2009) reports UK business spends 75% of the software development budget on fixing security flaws after delivering the product. This is a huge expenditure and it also creates untrustworthiness amongst customers. Figure 1 shows the chart for the annual spending on IT security which demonstrated the increase of more than 10% each year (Gartner 2016).

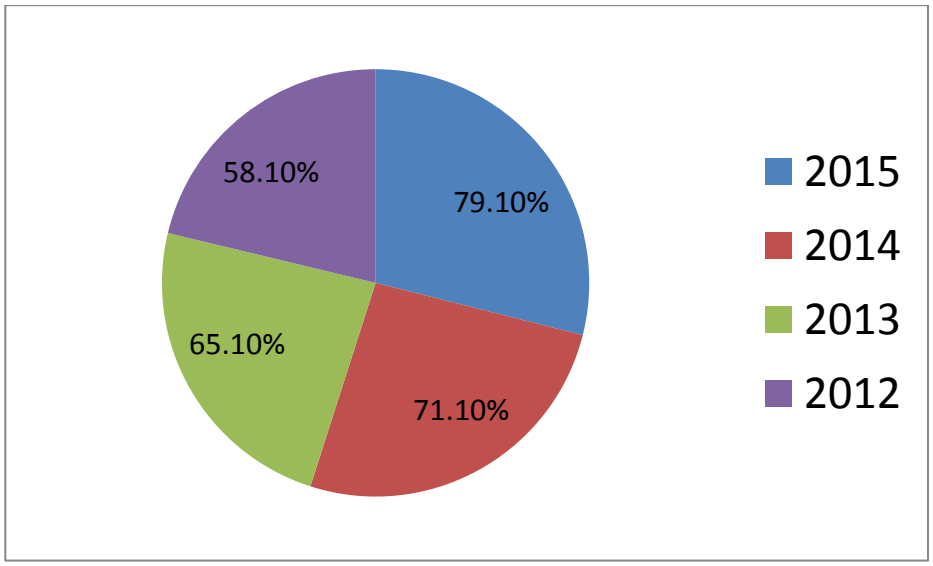

Figure 1 Annual Spending on Information Security

In addition, recent issues with cloud security strategies, techniques are elaborated by Chang and Ramachandran (2015;2016).

Allen et al. (2008) state that the one of the main goals of Software Security Engineering is to address software security best practices, process, techniques, and tools in every phases and activities of any standard software development life cycle (SDLC). The main goal of building secured software which is defect FSRee and better built with:

- Continue to operate normally in any event of attacks and to tolerate any failure
- Limiting damages emerging as an outcome of any attacks triggered
- Build Trust & Resiliency In (BTRI)
- Data and asset protection

In other words, secured software should operate normally in the event of any attacks. In addition, it involves the process of extracting security requirements from overall system requirements (includes hardware, software, business, marketing, and environmental requirements) and then also further refined and extracted security and software security requirements from software and business requirements. Then the refined software security requirements can be embedded and traced across the software development life cycle (SDLC) phases such as requirements, design, development, and testing. This has not explained well in security related literatures so far. This provides a clear definition of eliciting software security requirements.

Examples of recent cyber attacks include attacks on Carphone warehouse which resulted in the loss of personal details of up to 2.4 million Carphone Warehouse customers may have been accessed in a cyber-attack, the mobile phone retailer says. Similar news has been too FSRequent in recent news such as Talktalk, etc. Therefore, the conclusion is, what we have discovered is only a handful of cyber-attacks and software vulnerabilities and what is left to uncover is the size of the sea.

3 Software Security Requirements Engineering and Management

Requirements are the starting point, responsible for any system, legal and contractual issues, governance, and provide full functional perspective of the system being developed. Requirements Engineering is a discipline in its own right, which provides process, tools, techniques, modelling, cost estimation, project planning, and contractual agreements. There are wealth of requirements engineering methods, techniques, best practice guidelines, and tools (Jacobson 1992; Kotenya and Sommerville, 1992; Lambsweede 2009; Sawyer and Sommerville 1998). However, due to the nature of increased demands for security-driven applications, current techniques are inadequate for capturing security related requirements effectively. Firesmith (2007; 2003) reports that poor requirements are the main reasons for cost and schedule overruns, poor functionality and delivered systems that are never used. Requirements are classified into two major parts such as functional requirements which deal with the functionality of the system and non-functional requirements which deal with constraints, quality, data, standards, regulations, interfaces, performance, reliability, and other implementation requirements. Studies (Jacobson 1992; Kotenya and Sommerville 1992; Lambsweede 2009; Sawyer and sommerville 1998) have shown that requirements engineering defects cost 10 to 200 times to correct the system after implementation. Therefore, it is paramount to get the requirements correct, concise, and unambiguous.

Capturing business security requirements is a collaborative effort involves many stakeholders such as business analysts, software requirements engineer, software architect, and test managers. Security requirements should provide a clear set of security specific needs and expected behavior of a system. The main aim is to protect systems assets (data and files) and unauthorised access to the system from intentional attacks to the application software systems and other forms of internet based security attacks such as spam, denial of service, identity theft, viruses, and many other forms of intentional attacks that emerges every day. Security remains a software problem as the number of threats and vulnerabilities reported to CERT-SEI & CERT-UK (Computer Emergency Response team) of 2493% increase between 1997 (311 cases reported), 2006 (8064 cases reported), and as of 30th April 2015 (3192 cases reported).

In traditional RE, security requirements are considered to be a part of non-function properties and are considered an aspect of implementation strategies such as password protection, authentication, firewalls, virus detection, denial-of-service attacks, etc. Therefore, security needs to be considered as highly specific set of requirements for every functional requirements that has been identified, and has to be applied throughout the life cycle so that we can achieve build in security (BSI). *In addition, current RE methods have considered mostly what the system must do, but what the system must not do*. This is the key issue that will be considered when selecting RE methods for software security. Moreover, in Software security RE methods, there are more stakeholders than traditional RE methods have considered such as social engineers, security specialist, business process modeling experts, service computing specialists, and users. Often attackers look for defects in the system, not the system features and functionalities.

This section will look at the various methodologies for eliciting requirements for software security. Most common best practices are:

1. Eliciting and extracting requirements for software security explicitly
2. Prioritising security requirements
3. Risk assessment for security requirements
4. Design and implement security requirements

Before embarking on details of various methods for eliciting security requirements, we will try and understand three important basic concepts: use cases, misuse cases, and abuse cases. The use case has been an effective form of representing user requirements (user who interacts with the system is also known as the actor) visually (Jacobson 1992; Kotonya and Sommerville 1998; Lamsweerde 2009). This visual form represents a user requirements story is also known as scenario. Therefore, a use case = scenario (user story) + actors (who interacts with the system). The combination of a scenario and actors is known as a use case. However, the use case has not exploited much for representing non-functional aspects of the systems. Hence, is the reason for the emergence of new techniques such as misuse and abuse cases. Misuse case can help us to represent security requirements visually From the attacker's point of view (an example might be trying to login to the system with different password combination) whereas the abuse case should represent security requirements From a much stronger destruction aspects of

the system (an example might be trying to destroy by changing registration key, configuration and other DLL files). A number of techniques have emerged to address RE from an attacker's perspective:

- Attack patterns are similar to design patterns which has been designed to study attacks from destructive mode, Allen et al. (2008) and Build Security In (BSI).
- Misuse and abuse cases are a set of use cases From an attacker's perspective, McGraw (2006)
- Attack trees provide a formal mechanism for analysing and describing various ways in which attacks can happen From an attacker's perspective. Simply represent attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as leaf nodes, Schneier (1999;2000) and Ellison and Moore (2003)
- Microsoft SDL (SDL 2016 & TAM 2016) provides support on threat modelling which describes a set of security aspects by defining a set of possible security attacks. This has an integral part of Microsoft's SDL method, Howard and LeBlanc (2002)
- Building Security In (BSI) method (2013), process, design principles, and techniques provided by McGraw (2006) and others which is now officially supported by the US department of Homeland security. Some of the design principles include:
    - Correctness by Construction (CbyC)
    - Securing the Weakest Link
    - Defense in Depth
    - Failing Securely
    - Least Privilege
    - Separation of Privilege
    - Economy of Mechanism
    - Least Common Mechanism
    - Reluctance to Trust
    - Never Assuming that your Secrets are Safe
    - Complete Mediation
    - Psychological Acceptability
    - Promoting Privacy
- The SEI's (Software Engineering Institute) has identified a method known as SQUARE (Secure Quality Requirements Engineering) method described by Mead et al. (2008) which is to elicit and prioritise requirements and its consists of nine steps as follow:

    - Agree on definition
    - Identify security goals
    - Develop artifacts
    - Perform risk assessments
    - Select an elicitation technique
    - Elicit security requirements
    - Categorise security requirements
    - Prioritise security requirements
    - Inspect security requirements
    - Clear identification of requirements of the whole application system and extract security requirements. Interact with stakeholders to clarify security requirements and the technology they want to use, and cost implications.

- OCTAVE method by Caralli at al. (2007), Alberts and Dorofee (2002) and Woody and Alberts (2007) provides clear activities on security requirements:
    - Identify critical assets
    - Define security goals
    - Identify threats
    - Analyze risks
    - Define security requirements
- Other methods include CLASP (2006) and S-SDLC and have given detailed descriptions by Ramachandran (Ramachandran 2012).

Chen (2004) distinguishes the key difference between software security engineering with that of robustness for software safety engineering. Software security engineering deals with engineering approach to software development with an aim to engineer and implement security features whereas robustness deals with engineering software

for safety critical systems. Therefore, we need to identify, analyse, and incorporate security requirements as part of the functional requirements process. Belapurkar et al. (2009) have identified a list of some high-level areas for each security specific functional requirements as follows:

- Identification should address how a system recognises the actors/entities (humans or systems) interacting with the system.
- Authentication should address how a system validates the identity of entities
- Authorisation should address what privileges are to be set for an entity interacting with a system
- Non-repudiation should address how a system prevents entities from repudiating their interactions with the system functionality
- Integrity should address how a system protects information From any intentional or unintentional modifications and tampering
- Auditing should address how a system allows auditors to see the status of the security controls in place
- Privacy should address how a system prevents the unauthorised disclosure of sensitive information
- Availability should address how a system protects itself From intentional disruptions to service so that it is available to users when they need it.

Software security requirements are not only a set of constraints on the software systems, but they satisfy required governance and provides protection and trust. This means that we need far newer techniques such as attack patterns, misuse and abuse cases as part any requirements process. Tasks and activities associated with requirements engineering management for software security are shown in Figure 2.

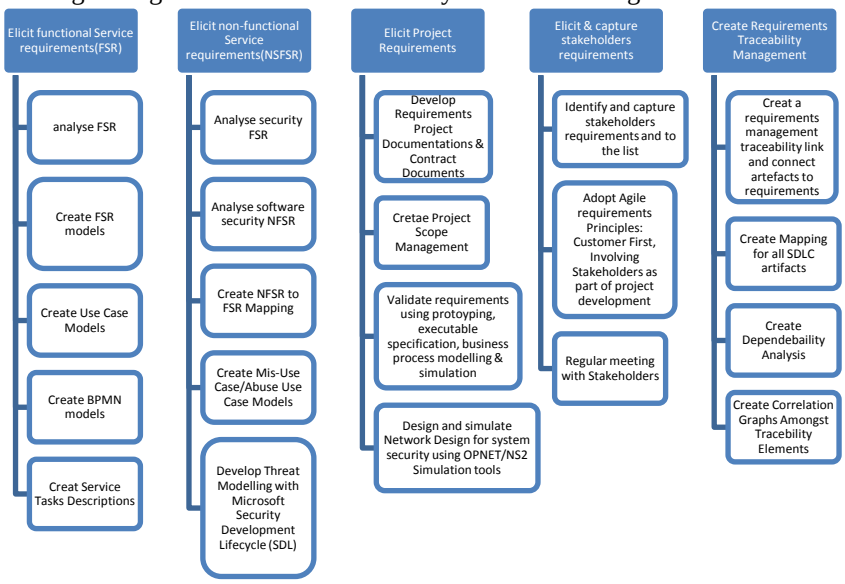

Figure 2 Requirements Management Tasks and Activities

As shown in Figure 2, requirements traceability management (RTM) been defined as the management of the requirements traceability matrix created and validated during requirements development phase. The RTM has been often created using a simple file or by the CASE tools. RTM links requirements artifacts across the other artifacts such as design, code, test case for a complete RTM. The issue is that existing traditional approaches have not considered software security that have been identified using current requirements, methods discussed in this section (misuse cases, threat modelling, etc.). Therefore the Figure 2 highlights the importance of integrating software security artifacts with the traditional RTM system. According our RTM method proposed consists of five tasks as part of the software security RE phase:

- Elicit functional service requirements (FSR) and the activities are analyse FSR, Create Functional model, Create Use Case Models
- Elicit non-functional service requirements (NFSR) and the associated activities are Analyse security FSR, Analyse software security NFSR, Create NFSR to FSR Mapping, Create Misuse Case and Abuse Case Models, Develop Threat Modelling with Microsoft Security Development Lifecycle (SDL)
- Elicit Project Requirements and the associated and linked activities are Develop Requirements Project Documentations & Contract Documents, Create Project Scope Management, Validate requirements

using protoype, executable specification, business process modelling & simulation, Design and simulate Network Design for system security using OPNET/NS2 Simulation tools

- Elicit & capture stakeholder requirements and the associated activities are Agile based such as: Identify and capture stakeholder requirements and to the list, Adopt Agile requirements Principles: Customer First, Involving Stakeholders as part of project development, Regular meeting with Stakeholders
- Create Requirements Traceability Management (RTM) and associated activities include Create a requirements management, traceability link and connect artifacts to requirements, Create mapping for all SDLC artifacts, Create Dependebaility Analysis, Create Correlation Graphs Amongst Traceability Elements

This key idea is to adopt a holistic approach to requirements engineering management for software security which includes state of the art techniques and practices. This will allow us to create a succinct software security requirements that are validated with simulation technologies such as BPMN (Business Process Modelling Notations) that allow validation of requirements by simulating the proposed processes against expected performance and the level of security with be builded-in (BSI).

4 Security Vulnerability Identification and Integration for DDoS Attacks

This section discusses on the number of vulnerabilities, also presents SDLC on threat modelling for vulnerability identification during RE phase. This is demonstrated with the use of DDoS (Distributed Denial-of-Service) Attacks which has been common in cloud services and other critical business sectors such as Carphone warehouse, talktalk mobile, etc. This is mostly because of the existing browsers that have been developed with BSI. Therefore, adding security through patches as seen in current practices has proven to be vulnerable to cybersecurity attacks. This is illustrated in Figure 3 on vulnerability analysis for various existing browsers using existing available data.

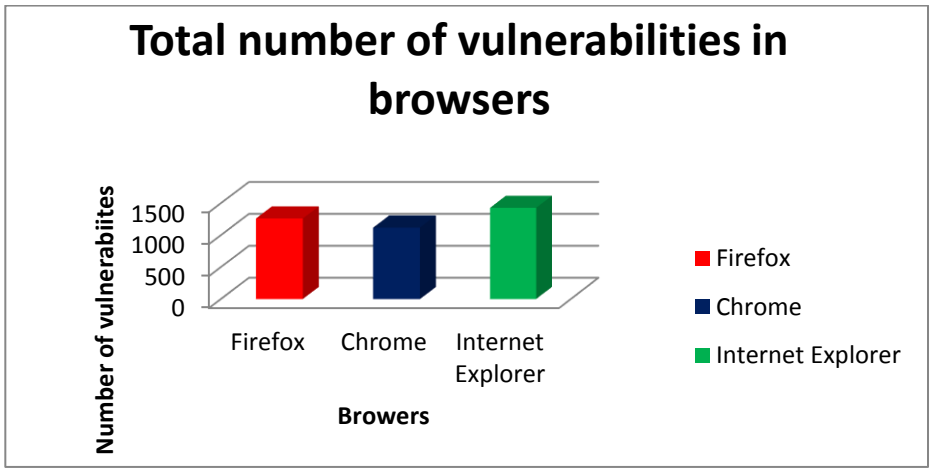

Figure 3 Known Vulnerability in Browsers

Risk analysis is also an important part of any requirements and modelling process. Ashbaugh (2006) says 90% of the software security vulnerabilities are caused by known software defect types and hence one way to reduce these vulnerabilities is to conduct risk analysis for software security throughout SDLC systematically.

In our approach, according our RTM methods, we propose the use of Microsoft Threat Modelling Tool known as Microsoft SDL (Security Development Lifecycle). This is shown in Figure 4 which consists of vision (software security requirements) leads to create threat models and allows us to consider various threats for each functional requirements and for each threat identified makes us to consider a number of mitigation. During this phase, a threat model report will be generated using the Microsoft threat modelling tool (TAM 2016). In our case study, we have used Amazon EC2 infrastructure and analysed for potential DDOS threat and mitigation.

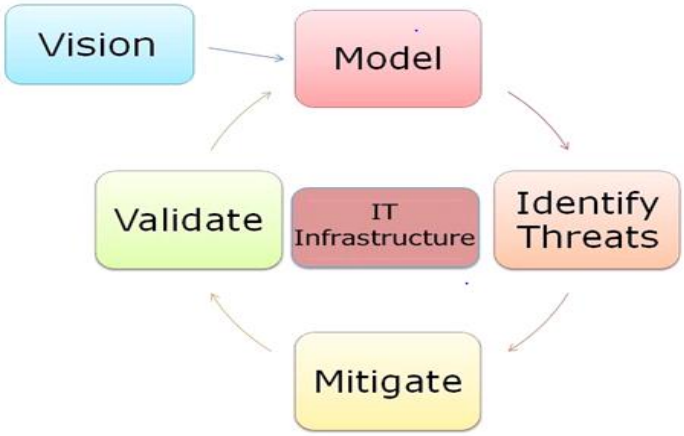

Figure 4 Microsoft Threat Modeling Process

The first step is building a clear vision by planning the location for each equipment. In this phase, it is preferred to produce a statement that explains the reason and the role of every asset chosen to be deployed, and this operation will help the organization to separate between activities.

After having a clear vision, a data flow diagram will be modeled in order to understand the flow of data between internal and external assets. Generally, there are four elements, which could be used for tagging boundaries:

1. Data flow: for tracking the flow of data from source to destination.
2. Entry points: is every asset that receives data from externals, and it can be a user, Hardware or software. At this level, we should eliminate DDoS potential attacks opportunities that may be exposed.
3. Trust Boundaries: which determines all trusted networks and domains; for example, devices such as routers, Firewall and load balancers.
4. Protected assets: which defines the critical assets, which must be, protected for instance customer's database or storage.

Because DDoS attacks are targeting only the availability of resources; **identifying threats** should focus only on the DDoS attack taxonomies (Resource and Bandwidth depletion) and then mark the severity of each of them.

The fourth step after threat identification is developing possible countermeasures for mitigating all identified threats. Different categories are available to mark all mitigation mechanisms such as; mitigated, not mitigated or do not require mitigation. In this stage, any proposed mechanism should be analyzed in depth by countering the attack from all dimensions.

The last step of the threat modeling process is the validation of all our analysis and findings. Furthermore, by measuring the impact of any attack in case of a mitigation failure; for example, if a DDoS attack has successfully targeted the internal load balancer; what are the consequences that may impact the resources?

As amazon AWS doesn't expose its internal architecture for security purposes. We have decided to rely on some research papers and online websites to better understand the architecture, and according to Amazon AWS reference architectures paper; different devices are deployed within its data centers realm, such as; Firewall, DNS Resolution, application and web servers, redundant multi zones, Elastic Load Balancers, Intrusion Prevention System (IPS), Amazon Simple Storage Service (S3), Amazon CloudFront. In order to organize the simulation; the project will be divided into two main phases:

The first phase is studying the behavior of Amazon EC2 architecture under DDoS attacks. In this phase two scenarios will be developed. The first one is generating a legitimate traffic From different resources, and then monitors the reaction of Amazon Instances. The second scenario will target these instances with different DDoS taxonomies and compare the results with our previous findings. The following Figure 5 DDoS attack tree is developed to direct our DDoS attack development.

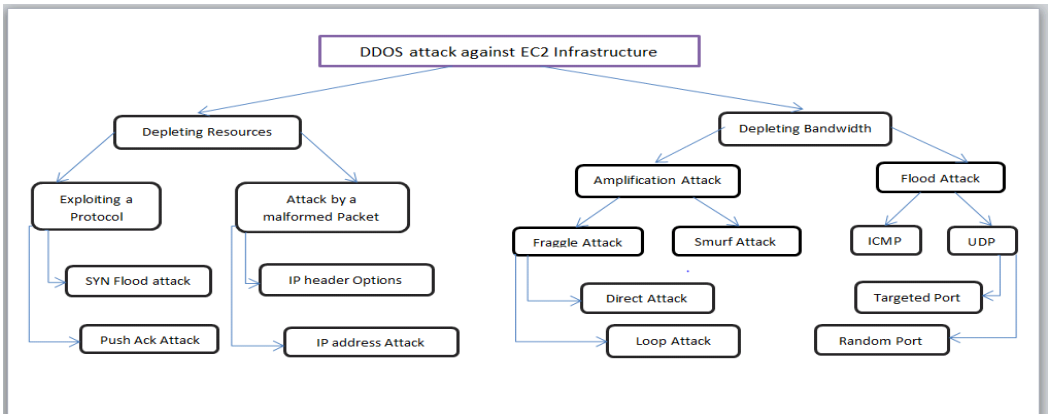

Figure 5 DDoS Attack Tree

Shneier (1999; 2000) has invented a methodical approach for describing threats against a protected, system. This process uses a tree structure for representing the objective as the root node, and leaf nodes will determine how to achieve that objective. Because of the high complexity of large scale networks; attack trees should receive high attention because if one idea is missed on how to compromise an asset; the attack tree will be useless. The above figure illustrates the various attacks that available to perpetrators for targeting Amazon AWS infrastructure by distributed denial of service attack using the resource or bandwidth depletion.

Analyzing the reasons behind the occurrence of DDoS attack will enable our project to develop a comprehensive mechanism that will counter broadly the current vulnerability of our systems and protocols.

It is necessary that our research should follow a systematic approach for the purpose of developing the product, and all selected methods and procedures should be suitable for the product requirements, also procedures should have high standards in term of efficiency and accuracy.

4.1 Identify threats

In this phase, threats can be identified and classified based on STRIDE model that was developed by Microsoft, and it consists of classifying threats into six categories: Spoofing, Tampering, Repudiation, denial of service and Elevation of privilege (STRIDE 2002). STRIDE approach consists of analyzing each component against potential threats using process decomposition, and for every threat, possible mitigations can be proposed allowing consolidation of robust security in cloud infrastructures. This is shown in Table 1.

Threat	Security Property
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-Repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

Table 1 Threats and Security Properties

As DDoS attacks target only the availability of resources, we will shrink our scope to only focus on a DOS category by identifying the assets that could be targeted as detailed below:

- Firewall and IPS are designed to address threats targeting network integrity and confidentiality by storing and inspecting each communication session stored on its entry table. During a DDoS; an attacker could exhaust their resources by filling the entry table with bogus packet information.
- As we know, server load balancers can be employed for tolerating server failure by distributing the task across different data centres for achieving high application availability as well as server scalability. However, the effectiveness of load balancing algorithms (Round Robin, Randomized, Threshold, Central Manager) is facing various challenges in term of throughput, response time scalability and resource utilization during a network flood as different research papers have evaluated static and dynamic load balancing algorithms performance.

- Links deployed within the cloud infrastructure can be tightened with layer 3/4 volumetric attack, especially when the Botnet is able to generate up to 300 GB traffic that is hard to be handled with the current available network links.

Ranking is part of threat identification process; any critical component that its failure could cause a serious bottleneck should be assigned the highest rank that later on during the mitigation process the component will receive more attention as well as advanced security controls. However, Microsoft DREAD threat risk ranking model could be employed to rank threats based on: damage potential, reproducibility, exploitability, affected users and discoverability. We have applied DREAD model to our cloud infrastructure scenario in order to come up some defense techniques against DDoS attacks, as shown in the Table 2.

Threat	D	R	E	A	D	Total	Rating
Exhausting the resource of a cloud server using a flood based Internet Group Management Protocol.	8	9	7	6	5	35	Medium risk
Targeting a cloud server with DNS amplification attack (Receiving high load of recursive queries).	9	9	9	9	9	45	Medium risk
A cloud server will be targeted with an ICMP amplification attack (Spoofing the source address of ping request).	4	7	8	3	4	26	Medium risk
An end system will be crashed by a Land attack (Receiving packets with same IP source and destination address).	4	7	8	3	4	26	Medium risk
Consuming the resources of a system using SYN flood attack.	4	7	8	3	4	26	Medium risk
A SIP Proxy server could be targeted by a high load of SIP invite messages.	8	8	7	7	7	37	High risk
The attacker congest the uplink with ICMP flood	9	9	9	9	9	45	High risk
The attacker congests the uplink with UDP flood	9	9	9	9	9	45	High risk
Crashing Cisco routers with an unexpected header (DOS attack).	9	9	9	9	9	45	High risk
Crashing an end system by ping of death attack (Sending oversized ping request packets which violates the Internet Protocol).	5	6	7	3	3	24	Medium risk
Crashing a firewall with unexpected header (DOS attack).	9	8	8	0	8	33	Medium risk
Overflowing the IPS/Firewall entry table with bogus packets.	9	8	8	0	8	33	Medium risk

Table 2 Threat Prioritization based on DREAD

D: damage potential **E:** exploitability **D:** discoverability **R:** reproducibility **A:** affected users

All threats that classified as high risk should receive extra control by building for each of them a robust mitigation process as well as an incident response plan. Alternatively, we can employ the following formula to perform threat risk assessment:

Risk= Threat * Vulnerability * Consequence

Attack tree is considered one of the most effective approaches for listing all opportunities available to perpetrators in order to achieve their objectives, and it can be done by thinking From the adversary's perspective on how to degrade the performance of a cloud infrastructure, or if it's possible crashing down available resources. We have built a DDoS attack tree to give a clear picture on different opportunities available to perpetrators, and it is not complete because every cloud infrastructure has its own design, equipment, services, security policies and network throughput. This should be done similar to the diagram shown in Figure 5.

4.2 Building Attack Tree and Mitigation Process

DDoS attacks are most common type of security attacks which has been frequently seen in recent years as discussed earlier. DDoS attacks can disable a complete feature and make resources inaccessible. DDoS attack is the biggest threat to cloud computing and for its success. Therefore, it is the key part of our approach to demonstrate systematic approach to security requirements can prevent DDoS influencing through design and development. Our approach is to use Microsoft threat modelling tool to build attack tree and use misuse cases to identify security requirements. We also propose to use business process, network, and cloud simulation tools to completely study and evaluate the proposed design. After building a data flow diagram; we can use the results provided by the attack tree to determine which countermeasures should be employed to consider each attack, and each countermeasure should be characterized by a status. DDoS state of art has complicated the task for security providers to build a comprehensive protection that can detect and discard DDoS traffic in real time without alteration with legitimate traffic. However, the following formula is developed to give insight about the best approach that can defeat completely this threat, and also it will direct cloud providers as well as enterprises to be ready for the attack before taking place. Therefore, we can define DDoS protection process as:

Security requirements with DDoS Protection= Contingency plan +DDoS security best practices + use misuse cases

When a DDoS attack is being detected by monitoring tools; the cloud provider should have an incident response (IR) plan, which details the processes that should be followed during mitigation. IR plan comprises: incident documentation, containment strategies, incident escalation and incident recovery. Some examples of incident response procedures are detailed below:

- Use specialised devices to stop the attack near the network cloud.
- Using black holing to direct DDoS traffic to a null port.
- Switching to alternate networks.
- Unwanted connections on routers should be terminated.
- Contacting the internal team for attack visibility (Security staff + Network operation).
- Relieving the pressure on infrastructure using upstream filtering.

Deshmukha and Devadkarb (2015) discuss that the main intention of a DDoS attack is to make the victim unable to use the resources. In most of the scenarios, targets could be web servers, CPU, Storage, and the other Network resources. In cloud environment also DDoS can reduce the performance of cloud services significantly by damaging the virtual servers.

5 Integrated Security Software Development Lifecycle Process

The above discussed drawbacks and requirements for a concise method, lead us to develop a model that integrates various activities of identifying and analysing software security engineering into the software development process, and this new process and its activities is shown in Figure 6. However, this paper focuses on only software

security requirements specific activities. According to this model, SSRE (software security requirements engineering) consists of identifying standards and strategies of the organisation with regards to requirements elicitation (including analysis, validation, verification), conducting risk management and mitigation, and identifying software security requirements consists of a further sub-processes of defining security, identifying security strategies, conducting areas and domain scope analysis, business process modeling and simulation, identifying security issues, applying use cases and misuse cases, attack patterns.

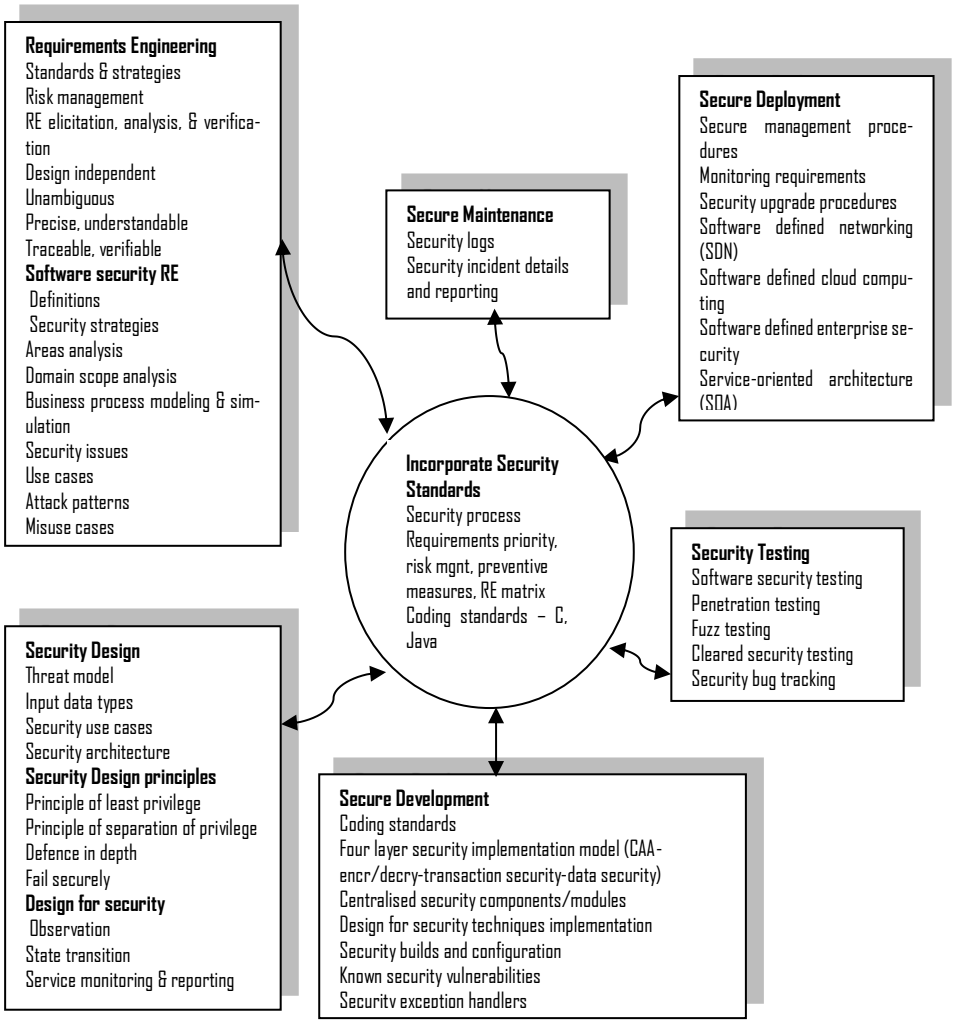

Figure 6 Integrated secure software development engineering life cycle (IS-SDLC)

Likewise, this model also provides security-specific processes for identifying security threats during design, development, testing, deployment, and maintenance. There are a numerous number of good design principles that can be found in a vast majority of software design literatures. However, the following is a list of some of the key design principles that are highly relevant to software security design and are part of our IS-SDLC model:

- Principles of least privilege states to allow only a minimal set of rights (privileges) to a subject that requests access to a resource. This helps to avoid intentional or intentional damage that can be caused to a resource in case of an attack.
- Principles of separation of privilege states that a system should not allow access to resources based on a single condition rather it should be based on multiple conditions which has to be abstracted into independent components.
- Design by incorporating known CVE
- Design for resilience for which we have team up with IBM to develop a resilience model which supports system sustainability along side with Building Trust and Security In (BTSI)
- Select software security requirements after performance simulation using BPMN (Business Process Modeling Notation) and is described in detail by Ramachandran (2014; Chang and Ramachandran 2016).

SSRE activities in our IS-SDLC supports security in software defined networking (SDN), Cloud computing services (Software as a service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), Enterprise security includes cloud service providers and service consumers, and design for security principles and techniques. This is one of our unique contributions to the body of knowledge in software security research.

6 Software Security Requirements Engineering Method as part of IS-SDLC

Secure software engineering has taken up widespread understanding of the importance of integrating security across development life-cycle phases. Ouselati et. al (2016) discusses the challenges of secure software development using Agile based approach. Similarly, during requirements engineering activities, existing methods discuss lots on requirements change and do not consider integrating security as part of requirements evolution. Mead, Morales and Alice (2015) have proposed to enhance current SDLC with misuse cases derived from malware analysis based on attacks. Beckers et al (2014) have proposed a structured, pattern-based method supporting eliciting security requirements and selecting security measures and their method guides potential cloud customers to model the application of their business case in a cloud computing context using a pattern-based approach, also known as Cloud System Analysis Pattern. However, these approaches are very interesting but lacks integrating security requirements from various stakeholders. This paper aims to fulfil that gap by integrating various stakeholders and they often provide differing software security requirements which needs to be consolidate by requirements engineers. Software development and secure software development involve many stakeholders and business leaders and their coordination is critical for delivering secure software systems. The various stakeholders is shown in the Figure 7.

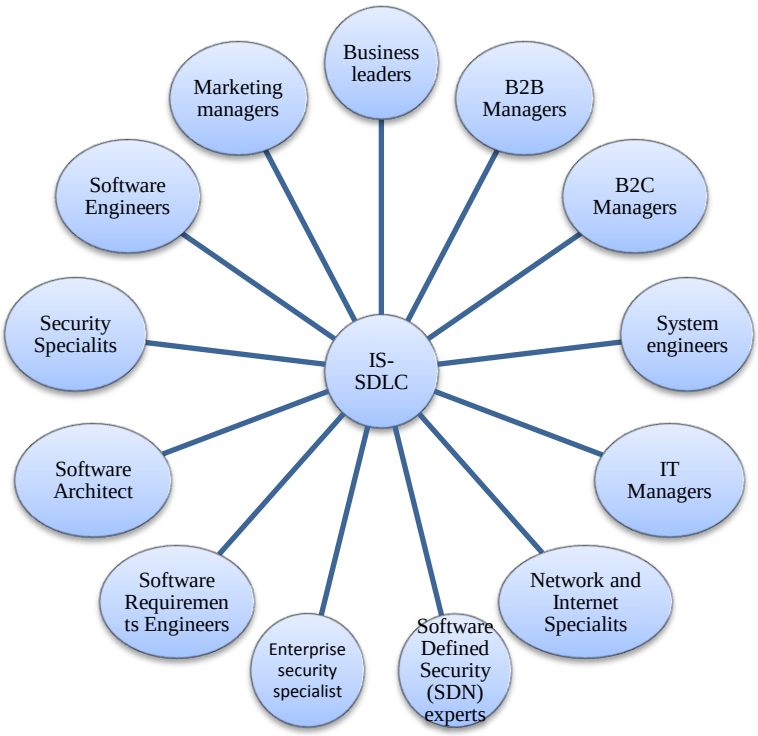

Figure 7. Stakeholders in Integrated Secure-SDLC (IS-SDLC)

The previous section has provided a brief account of various methodologies for eliciting requirements for software security. Most common best practices are:

1. Eliciting and extracting requirements for software security explicitly with visual notations
2. Prioritising software security requirements
3. Risk assessment and mitigation for software security requirements
4. Design and implement software security requirements
5. Providing SDLC life-cycle support

Existing methods lack heavily on incorporating social engineering to study software security requirements (learning From real experiences), security-specific business process modeling, performance simulations of the security-specific business processes, service computing, current and emerging technologies such as cloud computing, software-defined networking architecture, and software-defined enterprise security, and emerging vulnerabilities, and cyber attacks. This leads us to develop an integrated-secure software development model supporting software security requirements to be assessed and implemented explicitly in our method as presented in this paper. Ramachandran (2012) provides a comparative analysis of various software security requirements engineering (SSRE), design, and test methods based on our evaluation criteria used and this will help organization and engineers to choose appropriate method that is suitable for the system being developed. Based on the experience with IS-SDLC model in various projects, this paper has identified a set of best practice guidelines and recommendations on SSRE and SSE in general.

As part of the IS-SDLC, this paper have developed a development process model for cloud services with BSI (Build Security-In) across the cloud service development lifecycle. It is also well known best practice to eliciting and validating service level requirements early can save cost as much as 70% of the overall test and development costs. As shown in the diagram, the cloud development process model consists of a number of phases such as RE for cloud, conducting BPM modelling and specification (using BPMN 2 standard and BPEL), identifying and specifying SLAs, building software security in, designing services, and test and deploy. This is shown in Figure 8.

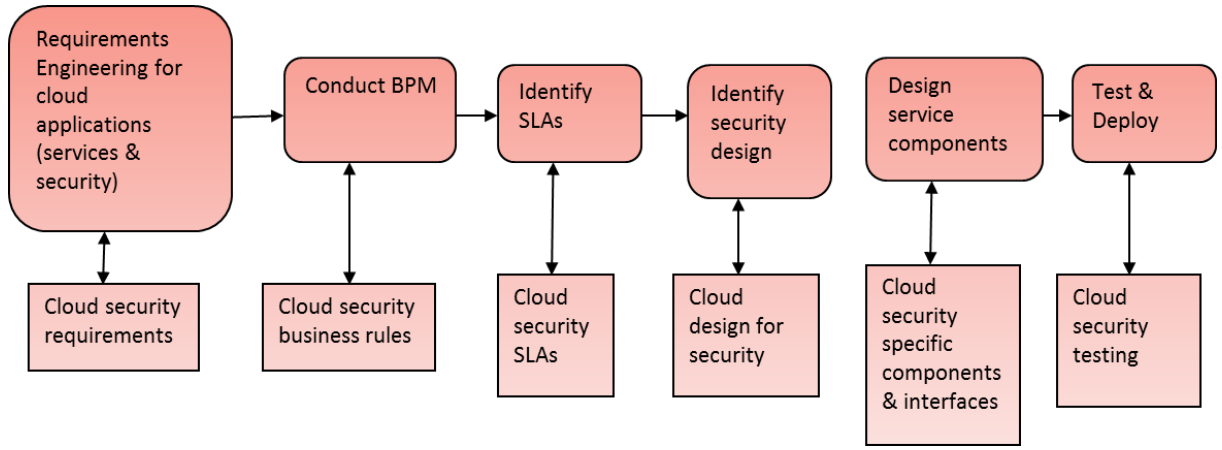

Figure 8 IS-SDLC's Cloud services development process

The RE phase consists of a number of parallel sequences such as clear identification of service requirements along with security requirements, and conduct/develop BPMN (business process modelling notation) for each service scenario which will also allow us to simulate and study their performance effectiveness before actual design and implementation begins. This is new for the traditional requirements engineering process. The next step in our IS-SDLC model proposes to identify clear SLAs (service level contract and governance for each cloud service. The following step is to develop a design for cloud security utilizing all well known design principles and to design services using SoaML or service component models using UML 2. We could also apply classical SDLC models such as waterfall, spiral, and others to capture functional and non-functional features for each SaaS services and web services. For example, to develop eHealth cloud services, IS-SDLC recommends classical formal specification methods. During the design stage, we could employ a set of good design rationales to compose SaaS as components. During business process modelling and service level specification, we can use BPMN for modelling and BPEL to specify service level workflows. These artifacts can then easily be transformed into a set of SaaS services. The model shown in this section is a security driven cloud development process. Hence, the main reason for identifying, specifying, and designing service level software security specific issues have been addressed across all phases of the development process.

7 Best Practice Guidelines

For secured systems, this paper identifies a set of common guidelines that are applicable to most of the secure software development:

1. Develop a list of security requirements checklists and classify them as: critical, medium, and moderate.
2. Bring in a requirements inspection team to conduct the security requirements validation process
3. Identify, elicit, analyse, and manage security requirements
4. Specify and model misuse cases and derive security requirements frorroomm misuse cases
5. Cross-check operational and functional requirements against security requirements
6. Establish an organisational security culture (e.g, check to make sure proper use of email systems do's and don'ts).
7. Apply DREAD model and Microsoft Security Development Lifecycle for identifying threat and mitigation
8. Apply business process Modelling and simulation using BPMN tools such as Bonita soft which provides clear performance attributed for all selected security-specific processes.
9. Develop design for cloud security
10. Design cloud services with SoaML or UML 2.0 Component model
11. Apply network simulation tools to study and identify network threats

8 Conclusion

Software security engineering offers several best practices, techniques, and methods to develop systems and services that are built for security, resiliency, sustainability. However, software security can not be just added after a system has been built and delivered to customers as seen in today's software applications. This paper provided concise techniques and best practice requirements guidelines on software security and also discussed an Integrated-Secure SDLC model (IS-SDLC), which will benefit practitioners, researchers, learners, and educators. This paper has demonstrated the application of Microsoft Security Development Lifecycle and has developed an integrated secure software development process model for cloud computing.

Acknowledgement

I would like to thank COMPLEXIS 2016 conference organisers for inviting me as a keynote speaker as this paper is the result of this talk on complex information systems and cloud computing and big data.

References

Alberts, C and Dorofee, A (2002) Managing Information Security Risks: The OCTAVE[SM] Approach, Addison Wesley

Allen, J. H., et al. (2008) Software security engineering: a guide for project managers, Addison Wesley, 2008

Ashbaugh, D. A (2006) Assessing Information Security Risks in the Software Development Life Cycle, September, www.stsc.hill.af.mil

Ashford, W (2009) On-demand service aims to cut cost of fixing software security flaws http://www.computerweekly.com/Articles/2009/07/14/236875/on-demand-service-aims-to-cut-cost-of-fixing-software-security.htm

Belapurkar, A., et al. (2009) Distributed system security: issues, processes and solutions, Wiley.

BSI (2013) Attack patterns articles, https://buildsecurityin.us-cert.gov/articles/knowledge/attack-patterns

Beckers, K., et al. (2014) A Structured Method for Security Requirements Elicitation concerning the Cloud Computing Domain, International Journal of Secure Software Engineering, 5(2), 20-43, April-June 2014

Caralli, R. A et al. (2007) Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process, TECHNICAL REPORT, CMU/SEI-2007-TR-012

CERT-SEI, www.cert.org

CERT-UK, https://www.cert.gov.uk/

Chang, V and Ramachandran, M (2016) Towards achieving Cloud Data Security with the Cloud Computing Adoption Framework, IEEE Transaction on Service Computing, February 2016

Chang, V, Ramachandran, M, Kuo, Y-H (2015) Cloud Computing Adoption Framework - a security Framework for business clouds, Future Generation Computer Systems, Elsevier, 4* Journal Impact Factor: 2.786, 19th October 2015, http://www.sciencedirect.com/science/article/pii/S0167739X15003118

Chen, A. Jia (2004) Security engineering for software (SES), CS996-CISM, isis.poly.edu/courses/cs996-management/Lectures/SES.pdf

CLASP (2006) OWASP CLASP Version 1.2, http://www.lulu.com/items/volume_62/1401000/1401307/3/print/OWASP_CLASP_v1.2_for_print_LULU.pdf
Deshmukha, R. V., and Devadkarb, K. K (2015) Understanding DDoS Attack & its Effect in Cloud Environment, Procedia Computer Science 49 (2015) 202 – 210
Ellison, R.J. and Moore, A. P (2003) Trustworthy Refinement Through Intrusion-Aware Design (CMU/SEI-2003-TR-002, ADA414865). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.
Firesmith, D (2003) Engineering security requirements, Journal of Object Technology, Volume 2, No. 1, 2003
Firesmith, D (2007) Engineering Safety- & Security-Related Requirements ICCBSS Tutorial, SEI, Carnegie Mellon University, 27 February.
Gartner (2016) http://www.gartner.com/newsroom/id/2828722, [accessed on 15th March 2016]
Howard, M and LeBlanc, D. C (2002) Writing Secure Code (2nd ed.). Redmond, WA: Microsoft Press.
Han, G, et al. (2016) An Efficient Virtual Machine Consolidation Scheme for Multimedia Cloud Computing, www.mdpi.com/journal/sensors, Sensors 2016, 16, 246
Hsu, I-C and Cheng, F-Q (2015) SAaaS: a cloud computing service model using semanticbased agent, Expert Systems, February 2015, Vol. 32, No. 1, DOI: 10.1111/exsy.12063
Jacobson, I (1992) Object oriented software engineering: use case driven approach, Addison Wesley
Kotonya, G and Sommerville, I (1998) Requirements Engineering: Processes and Techniques, Wiley.
Kostoska, M, Gusev, M, and Ristov, S (2014) A New Cloud Services Portability Platform, 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, Vienna, 2013
Lamsweerde, van A (2009) Requirements Engineering: From system goals to UML models to software specifications, Wiley, UK.
McGraw, G (2006) Software security: building security in, Addison Wesley, USA
Mead, N. R et al. (2008) Incorporating Security Quality Requirements Engineering (SQUARE) into Standard Life-Cycle Models, SEI Technical Note CMU/SEI-2008-TN-006, http://www.sei.cmu.edu
Mead, N. R., Morales, A, J., and Alice, P. G (2015) A Method and Case Study for Using Malware Analysis to Improve Security Requirements, Intl. J of secure software engineering, Volume 6, Issue 1, IGI Global
Oueslati, H., et. al. (2016) Evaluation of the Challenges of Developing Secure Software Using the Agile Approach, Intl. J of secure software engineering, Volume 7, Issue 1, IGI Global
Ramachandran, M (2012) Software Security Engineering: Design and Applications, Nova Science Publishers, New York, USA, 2012. ISBN: 978-1-61470-128-6, https://www.novapublishers.com/catalog/product_info.php?products_id=26331
Ramachandran, M (2014) Enterprise Security Framework for Cloud Data Security, Book chapter "Delivery and Adoption of Cloud Computing Services in Contemporary Organizations, Chang, V (ed.) IGI GlobalGraham, D (2006) Building Security In, https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/requirements/548-BSI.html
Schneier, B (1999) Attack Trees: modelling security threats, Dr Dobbs Journal, December, http://www.schneier.com/paper-attacktrees-ddj-ft.html
Schneier, B (2000) Secrets and Lies: Digital Security in a Networked World. New York, NY: John Wiley & Sons
SDL (2016) Microsoft Security Development Lifecycle, https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx
Sommerville, I and Sawyer, P (1998) Requirements Engineering: A good practice guide, Wiley.
S-SDLC: Introducing Secure Software development Life Cycle (S-SDLC), Infosec Institute, http://resources.infosecinstitute.com/intro-secure-software-development-life-cycle/
STRIDE (2002) The STRIDE Threat Model, https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx
Srivastava, S et al. (2015) The Technological Growth in eHealth Services, Computational and Mathematical Methods in Medicine, Volume 2015, Article ID 894171, 18 pages, http://dx.doi.org/10.1155/2015/894171, Hindawi
TAM (2016) Mocrosoft Threat Modelling Tool, https://www.microsoft.com/en-us/download/confirmation.aspx?id=49168
Xu, X (2013) Cloud manufacturing: A new paradigm for manufacturing businesses, Australian Journal of Multi-Disciplinary Engineering, Vol. 9, No. 2, pp. 105-116, http://dx.doi.org/10.7158/N13-GC08.2013.9.2
Woody, C and Alberts, C (2007) Considering Operational Security Risk during System Development", IEEE Security & Privacy, pp. 30-43
Whiting, R (2015) 10 Emerging Cloud Vendors You Need To Know About, July 2015, http://www.crn.com/slideshows/cloud/300077581/10-emerging-cloud-vendors-you-need-to-know-about.htm/pgno/0/1